Simulation of a 1D Super-sonic nozzle flow simulation using Macromack method

Introduction :

Nozzles are most commonly used in every field in order to study the liquid and air properties, because those two element plays a major role in engineering fields. However we may have many properties and the behaviour of fluids and air but we are still in position to find the exact solutions for the equations which may help to find out the answers for the unknown questions.

The project which on based on nozzle is an important project to understand the working of a fluid particle when it is travelling through the **con-di nozzle**.

In this project we are using ***MacCormack method*** to solve the equations becuase this method is most widely used by engineers to solve the complex second order differtiation. This method provides **second order accuracy** without the use of **second order derivative**, so obviously the complexity of the equation is redused.

Objective :

The main objective of this project is to learn the fluid properties when it is flowing throught a con-di nozzle .

As mentioned in the question there are two types of flows namely :

- ***Conservative Flow***
- ***Non-Conservative Flow***

Conservative Flow :

In this conservative flow , when a fluid is flowing thorugh a pipe a small control volume is taken in some area inside that pipe and this control volume is fixed, so that the equation can be solved to find out the properties of the flow thorugh that point , this is known as Conservative flow.

Non-Conservative Flow :

In this Non-Conservative flow , a small control volume is taken inside the pipe flow but with a small change, the control volume here is moving with the flow instead of being fixed as we saw in Conservative Flow.

Programming concepts :

We use ***MacCormack method*** for solving the equation in which the concept is used in programming to give the exact solution. We use two kinds of programming methods in which these two methods are compared to find out which method produce the answer exactly and quickly. These two methods are going to solve the same equation in-order to find which method is suitable for the solution.

Initially the grid points are chosen as minumum value and then the grid points are increased to see the changes in the solutions.

At frist the **grid points** are taken as **31** because this is the maximum value for the steady state to attain. At this points the graphs are plotted and the result is given below.

Secondly the grid points is made double **(61)** to check whether there is a drastic change or not. These graphs are also plotted and shown below.

In a state both provide similar plots after the steady state is achieved. Nearly 1000 time steps are followed by time marching teqnique to provide a steady state. Now lets see the time wise variation along throat side.

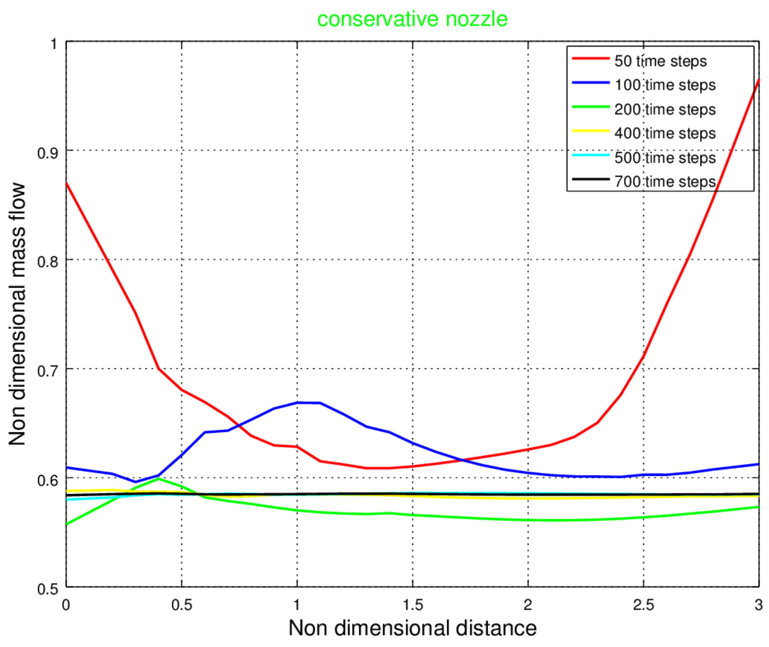
Time variation along the throat :

- *The concept we use in this problem is one dimensional method. The whole problem is assumed to be a one dimensional material in which a point is considered to read the properties over that point.*
- *But in reality there is always a two dimensional concept which flows through the nozzle.*
- *By taking all these constraints in the program we are provided with the results to find out the exact solution.*
- *As we see that Non-Conservative Flow provides more accurate results for primitve variables. This is because in Non-Conservative Flow the equations are direct subtitution* of *the primitve variables.*
- *But in Conservative Flow we use many different substitution for solving the same.*
- *As a result the Non-Conservative method provide less number of error when compared to Conservative solutions.*

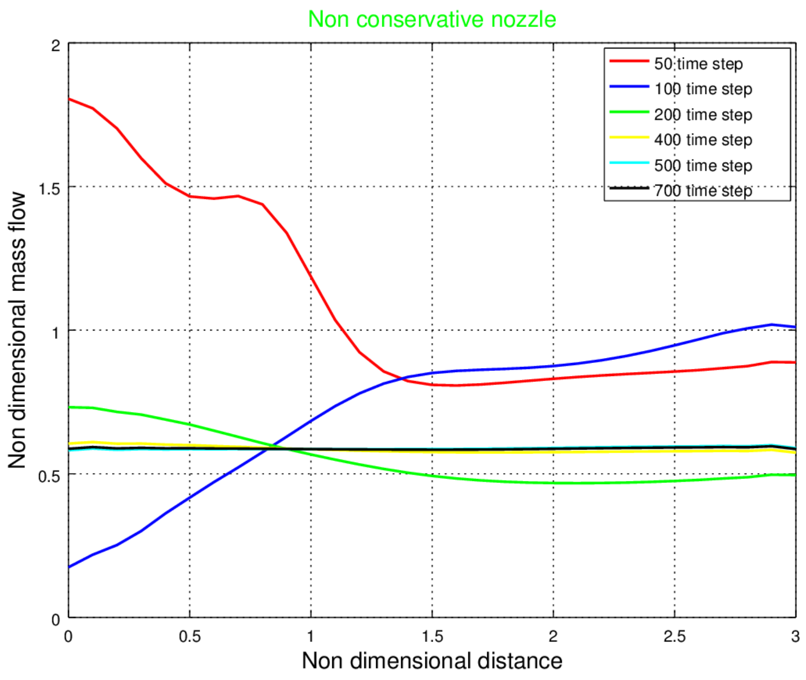
Now inorder to compare the values we are assuming different grid size , as we double the grid size the time steps should also be doubled inorder to provide an exact solutions for both the conservative and Non-conservative solutions.

Now lets see the output graphs of different parameters.

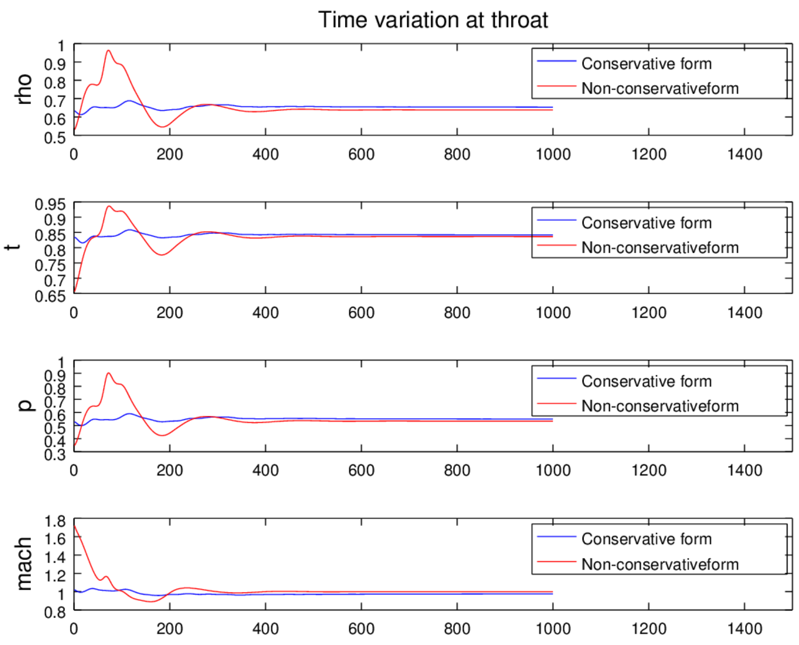
- ***Mass flow rate for Conservative Flow :***



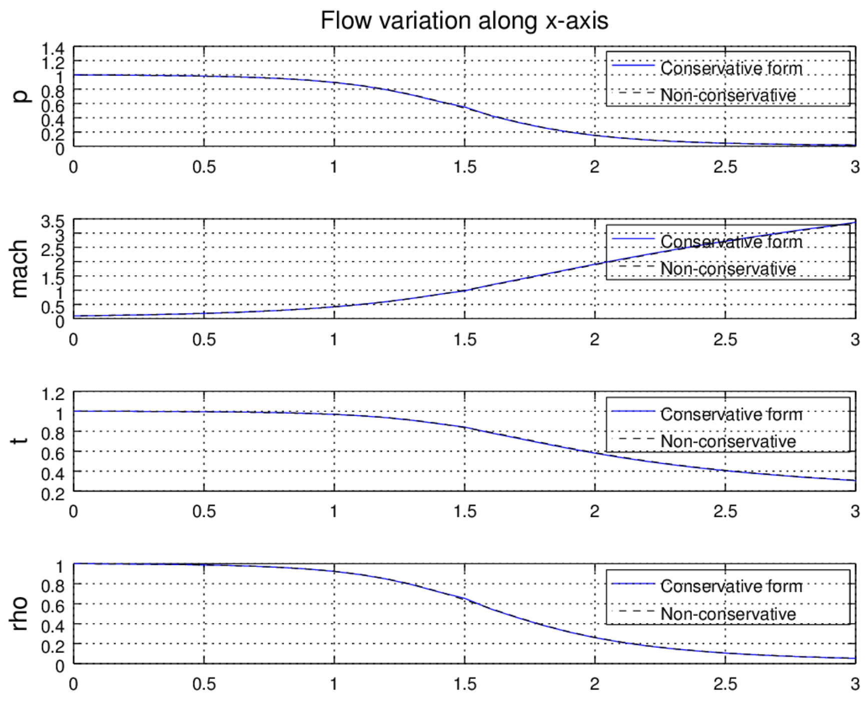
- ***Mass flow rate for Non-Conservative Flow :***



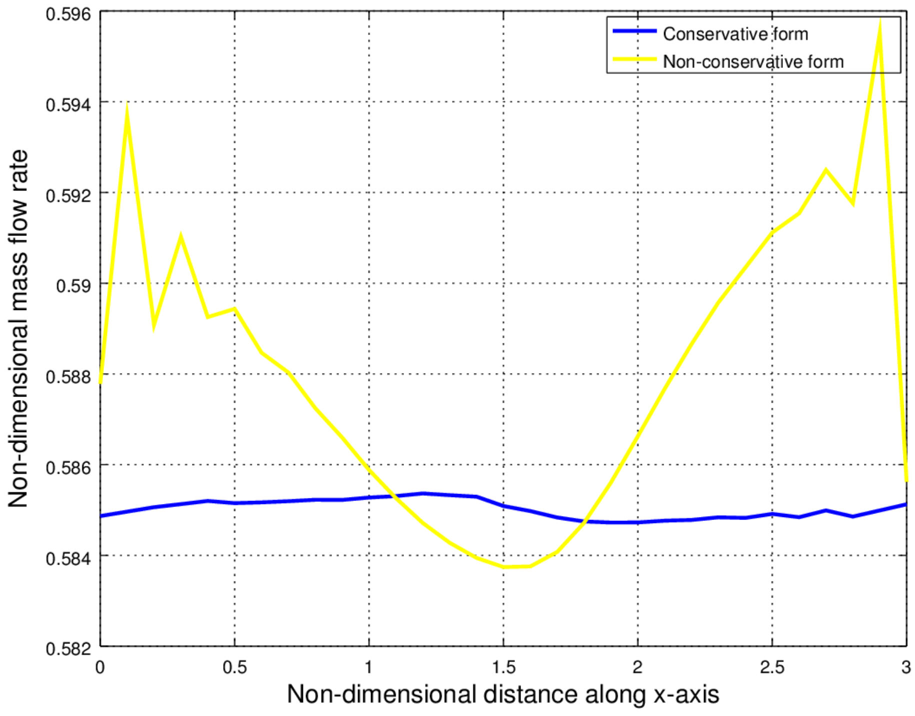
- ***Time variation at Throat :***



- ***Flow variation along X-axis :***



- ***Mass flow rate comparison :***



Now as we see the above graphs clearly depicts the Conservative and Non-Conservative graphs. These graphs are results of 31 grid points .

As we can see the conservative mass flow chart provide more stable results when compared to Non-conservative mass flow , because the mass flow is one of the dependent variables in the conservative form.

So as a result the Conservative form provide more accurate results for flux variables. so they are mostly used for Conservative flow.

Program for Conservative and Non-Conservative flow :

- ***Conservative Nozzle Flow Program :***

```
function [rho,t,m,mach,v,p,rho_t,v_t,t_t,mach_t,m_t,p_t] =
Conservative_nozzle_program(n,nt,trt)
```

```
% Now providing the initial conditions.

x = linspace(0,3,n);
dx = x(2)-x(1);
gamma = 1.4;

% Area initializing.
a = 1+2.2*(x-1.5).^2;
c = 0.5

% Now providing the concervative intial conditions for the remaining datas.

for i = 1:n
    if (x(i)>=0 && x(i)<=0.5)
        t(i) = 1;
        rho(i) = 1;
    elseif (x(i)>=0.5 && x(i)<=1.5)
         t(i) = 1-0.167*(x(i)-0.5);
         rho(i) = 1-0.366*(x(i)-0.5);
     elseif (x(i)>=1.5 && x(i)<=3.5)
          t(i) = 0.833-0.3507*(x(i)-1.5);
          rho(i) = 0.634-0.3879*(x(i)-1.5);
 end
 end


v = 0.59./(rho.*a);

% Now initializing the time step.

dt = min (c.*(dx./((t.^0.5)+v)));

% Now providing the solution vectors.

p = rho.*t;
u_1 = rho.*a;
u_2 = rho.*a.*v;
u_3 = u_1.*((t./(gamma-1))+((gamma/2).*(v.^2)));

for k = 1:nt

 u_1old = u_1;
 u_2old = u_2;
 u_3old = u_3;

 % Now in this conservative form a new term called flux term was introduce and it
should be also intialised for the problem.

f_1 = u_2;
term = (((gamma-1)/gamma)*(u_3-((gamma/2)*((u_2.^2)./u_1))));
f_2 = ((u_2.^2)./u_1)+ term;
f_3 = ((gamma*u_2.*u_3)./u_1)-(gamma*(gamma-1)*(u_2.^3))./(2*u_1.^2);

% predictor method.

  for j = 2:n-1

    df_1dx(j) = (f_1(j+1)-f_1(j))/dx;
```

```
   df_2dx(j) = (f_2(j+1)-f_2(j))/dx;
   df_3dx(j) = (f_3(j+1)-f_3(j))/dx;
   dlog_a_dx(j) = (log(a(j+1))-log(a(j)))/dx;

   dadx(j) = (a(j+1)-a(j))/dx;
   j_2(j) = (1/gamma)*(rho(j)*t(j))*dadx(j);

   % Now sustituting the values in the eqautions.

   % continuity equation

    du_1dt_p(j) = -df_1dx(j);

    % Momentum equation.

    du_2dt_p(j) = -df_2dx(j) + j_2(j);

    % Energy equation.

     du_3dt_p(j) = -df_3dx(j);

     % Now providing solution update.

      u_1(j) = u_1(j) + du_1dt_p(j)*dt;
      u_2(j) = u_2(j) + du_2dt_p(j)*dt;
      u_3(j) = u_3(j) + du_3dt_p(j)*dt;
 end

rho = u_1./a;
v = u_2./u_1;
t = (gamma-1)*((u_3./u_1)-((gamma*v.^2)/2));

f_1 = u_2;
term_1 = (((gamma-1)/gamma)*(u_3-(gamma*u_2.^2)./(2*u_1)));
f_2 = ((u_2.^2)./u_1)+ term_1;
f_3 = ((gamma*u_2.*u_3)./u_1)-(gamma*(gamma-1)*(u_2.^3))./(2*u_1.^2);

% Corrector method.

  for j = 2:n-1

   df_1dx(j) = (f_1(j)-f_1(j-1))/dx;
   df_2dx(j) = (f_2(j)-f_2(j-1))/dx;
   df_3dx(j) = (f_3(j)-f_3(j-1))/dx;
   dlog_a_dx(j) = (log(a(j))-log(a(j-1)))/dx;

   dadx_1(j) = (a(j)-a(j-1))/dx;
   j_3(j) = (1/gamma)*(rho(j)*t(j))*dadx_1(j);

    % Now sustituting the values in the eqautions.

    % continuity equation.

    du_1dt_c(j) = -df_1dx(j);

    % Momentum equation.

    du_2dt_c(j) = -df_2dx(j) + j_3(j);
    % Energy equation.

    du_3dt_c(j) = -df_3dx(j);

 end

  % Now computing the average time derivative.

  du_1dt = 0.5*(du_1dt_p + du_1dt_c);
  du_2dt = 0.5*(du_2dt_p + du_2dt_c);
  du_3dt = 0.5*(du_3dt_p + du_3dt_c);

  % Final solution update.
   for i = 2:n-1
       u_1(i) = u_1old(i) + du_1dt(i)*dt;
       u_2(i) = u_2old(i) + du_2dt(i)*dt;
       u_3(i) = u_3old(i) + du_3dt(i)*dt;
    end

  % Applying boundary condition.

  % Inlet condition.

  u_1(1) = rho(1)*a(1);
  u_2(1) = 2*u_2(2) - u_2(3);
  v(1) = u_2(1)/u_1(1);
  u_3(1) = u_1(1)*((t(1)/(gamma-1))+((gamma/2)*v(1)^2));

  % Outlet condition.

  u_1(n) = 2*u_1(n-1) - u_1(n-2);
  u_2(n) = 2*u_2(n-1) - u_2(n-2);
  u_3(n) = 2*u_3(n-1) - u_3(n-2);

  rho = u_1./a;
  v = u_2./u_1;
  t = (gamma-1)*((u_3./u_1)-((gamma*v.^2)/2));


  % calculating solutions
  mach = v./((t).^0.5);
  m = rho.*v.*a;
  p = rho.*t;

  % Throat.

  rho_t(k) = rho(trt);
   p_t(k) = p(trt);
   t_t(k) = t(trt);
   mach_t(k) = mach(trt);
   v_t(k)= v(trt);
   m_t(k) = m(trt);

   % Now plotting the mass flow rate at different time step.
  figure(1)
   if k == 50
      plot(x, m, 'r', 'linewidth', 1.5)
    hold on
      elseif k == 100
```

```
        plot(x, m, 'b', 'linewidth', 1.5)
      hold on
      elseif k == 200
        plot(x, m, 'g', 'linewidth', 1.5)
      hold on
        elseif k == 400
        plot(x, m, 'y', 'linewidth', 1.5)
      hold on
        elseif k == 500
        plot(x, m, 'c', 'linewidth', 1.5)
      hold on
        elseif k == 700
        plot(x, m, 'k', 'linewidth', 1.5)

         grid on
       xlabel('Non dimensional distance','fontsize',14);
       ylabel('Non dimensional mass flow', 'fontsize',14);
       title('conservative nozzle', 'fontsize',14, 'color', 'g');
       legend({'50 time steps' , '100 time steps' , '200 time steps' , '400 time
steps' , '500 time steps' , '700 time steps'});

    end

   end
```

- ***Non-Conservative Nozzle Program :***

```
function [rho,t,m,mach,v,p,rho_t,v_t,t_t,mach_t,m_t,p_t] =
Non_conservative_nozzle_program(n,nt,trt)

% Now providing the initial conditions.

x = linspace(0,3,n);
dx = x(2)-x(1);
gamma = 1.4;

% calculate initial profile.

rho = 1-0.3146*x;
t = 1-0.2314*x;
v = (0.1+1.09*x).*t.^0.5;

% Area initializing.0
a = 1+2.2*(x-1.5).^2;
% Now intializing the time step.

c = 0.5;


dt = min (c.*(dx./((t.^0.5)+v)));
```

```
for k = 1:nt

 rho_old = rho;
 v_old = v;
 t_old = t;

 % precdictor method.


   for j = 2:n-1

    dvdx = (v(j+1)-v(j))/dx;
    drhodx = (rho(j+1)-rho(j))/dx;
    dtdx = (t(j+1)-t(j))/dx;
    dlog_a_dx = (log(a(j+1))-log(a(j)))/dx;

    % Now sustituting the values in the eqautions.

   % continuity equation.

    drhodt_p(j) = -rho(j)*dvdx -rho(j)*v(j)*dlog_a_dx -v(j)*drhodx;

    % Momentum equation.

    dvdt_p(j) = -v(j)*dvdx -(1/gamma)*(dtdx + (t(j)/rho(j))*drhodx);

    % Energy equation.

    dtdt_p(j) = -v(j)*dtdx -(gamma-1)*t(j)*(dvdx + v(j)*dlog_a_dx);

     % Now providing solution update.

     rho(j) = rho(j) + drhodt_p(j)*dt;
     v(j) = v(j) + dvdt_p(j)*dt;
     t(j) = t(j) + dtdt_p(j)*dt;
   end
   % Corrector method.
    for j = 2:n-1

    dvdx = (v(j)-v(j-1))/dx;
    drhodx = (rho(j)-rho(j-1))/dx;
    dtdx = (t(j)-t(j-1))/dx;
    dlog_a_dx = (log(a(j))-log(a(j-1)))/dx;

    % Now sustituting the values in the eqautions.

   % continuity equation.

    drhodt_c(j) = -rho(j)*dvdx -rho(j)*v(j)*dlog_a_dx -v(j)*drhodx;

    % Momentum equation.

    dvdt_c(j) = -v(j)*dvdx -(1/gamma)*(dtdx + (t(j)/rho(j))*drhodx);

    % Energy equation.

    dtdt_c(j) = -v(j)*dtdx -(gamma-1)*t(j)*(dvdx + v(j)*dlog_a_dx);
```

```
end

% Now computing the average time derivative.

drhodt = 0.5*(drhodt_p + drhodt_c);
dvdt = 0.5*(dvdt_p + dvdt_c);
dtdt = 0.5*(dtdt_p + dtdt_c);

% Final solution update.

 for i = 2:n-1
     rho(i) = rho_old(i) + drhodt(i)*dt;
     v(i) = v_old(i) + dvdt(i)*dt;
     t(i) = t_old(i) + dtdt(i)*dt;
 end

 % Applying boundary condition.

 % Inlet condition.

 v(1) = 2*v(2)-v(3);

 % Outlet condition.

 rho(n) = 2*rho(n-1) - rho(n-2);
 v(n) = 2*v(n-1) - v(n-2);
 t(n) = 2*t(n-1) - t(n-2);

 % calculating solutions
 mach = v./((t).^0.5);
 m = rho.*v.*a;
 p = rho.*t;

 % Throat

 rho_t(k) = rho(trt);
 p_t(k) = p(trt);
 t_t(k) = t(trt);
 mach_t(k) = mach(trt);
 v_t(k)= v(trt);
 m_t(k) = m(trt);

 % Now plotting the mass flow rate at different time step.
 figure(2)
 if k == 50
    plot(x, m, 'r', 'linewidth', 1.5)
  hold on
    elseif k == 100
    plot(x, m, 'b', 'linewidth', 1.5)
  hold on
  elseif k == 200
    plot(x, m, 'g', 'linewidth', 1.5)
  hold on
    elseif k == 400
    plot(x, m, 'y', 'linewidth', 1.5)
  hold on
    elseif k == 500
    plot(x, m, 'c', 'linewidth', 1.5)
```

```
     hold on
       elseif k == 700
       plot(x, m, 'k', 'linewidth', 1.5)

       grid on
      xlabel('Non dimensional distance','fontsize',14);
      ylabel('Non dimensional mass flow', 'fontsize',14);
      title('Non conservative nozzle', 'fontsize',14, 'color', 'g');
      legend({'50 time step', '100 time step', '200 time step', '400 time step',
'500 time step', '700 time step'});
    end

   end
```

- ***Nozzle Main Function Program :***

```
clear all
close all
clc

% Now providing the initial conditions.
 n = 31;
 x = linspace(0,3,n);

 % Now providing the node values.
 trt = ((n-1)/2)+1;

% Now providing the time steps.

nt = 1000;

tic

 % Now providing the function of concervative program.
[rho2,t2,m2,mach2,v2,p2,rho_t2,v_t2,t_t2,mach_t2,m_t,p_t2] =
Conservative_nozzle_program(n,nt,trt);

time = toc

tic

% Now providing the function of Non_conservative program.
 [rho,t,m,mach,v,p,rho_t,v_t,t_t,mach_t,m_t,p_t] =
Non_conservative_nozzle_program(n,nt,trt);

time = toc
```

```
% ploting information along throat.
figure(3)
a1 = subplot(4,1,1);
plot(linspace(1,nt,nt),rho_t2);
hold on
plot(linspace(1,nt,nt),rho_t, 'color', 'r');
hold on
ylabel("rho",'fontsize',14);
legend(a1,{'Conservative form' , 'Non-conservativeform'});
axis([0 1500]);
title('Time variation at throat' , 'fontsize' , 14);


a2 = subplot(4,1,2);
plot(linspace(1,nt,nt),t_t2);
hold on
plot(linspace(1,nt,nt),t_t, 'color', 'r');
hold on
ylabel("t",'fontsize',14);
legend(a2,{'Conservative form' , 'Non-conservativeform'});
axis([0 1500]);



a3 = subplot(4,1,3);
plot(linspace(1,nt,nt),p_t2);
hold on
plot(linspace(1,nt,nt),p_t, 'color', 'r');
hold on
ylabel("p",'fontsize',14);
legend(a3,{'Conservative form' , 'Non-conservativeform'});
axis([0 1500]);



a4 = subplot(4,1,4);
plot(linspace(1,nt,nt),mach_t2);
hold on
plot(linspace(1,nt,nt),mach_t, 'color', 'r');
hold on
ylabel("mach",'fontsize',14);
legend(a4,{'Conservative form' , 'Non-conservativeform'});
axis([0 1500]);


% Plotting information along x axis.
figure(4)
b1 = subplot(4,1,1);
plot(x,p2);
hold on
plot(x,p,'--','color','k','markersize',4);
grid on;
ylabel("p",'fontsize',14);
legend(b1,{'Conservative form' , 'Non-conservative'});
title('Flow variation along x-axis' , 'fontsize' , 14);


b2 = subplot(4,1,2);
```

```
plot(x,mach2);
hold on
plot(x,mach,'--','color','k','markersize',4);
grid on;
ylabel("mach",'fontsize',14);
legend(b2,{'Conservative form' , 'Non-conservative'});


b3 = subplot(4,1,3);
plot(x,t2);
hold on
plot(x,t,'--','color','k','markersize',4);
grid on;
ylabel("t",'fontsize',14);
legend(b3,{'Conservative form' , 'Non-conservative'});


b4 = subplot(4,1,4);
plot(x,rho2);
hold on
plot(x,rho,'--','color','k','markersize',4);
grid on;
ylabel("rho",'fontsize',14);
legend(b4,{'Conservative form' , 'Non-conservative'});


% Mass flow rate comaprison.
figure(5)
plot(x,m2,'b', 'linewidth' , 2)
hold on
plot(x,m,'y', 'linewidth' , 2)
hold on
ylabel('Non-dimensional mass flow rate','fontsize', 14);
xlabel('Non-dimensional distance along x-axis','fontsize' , 14);
legend({'Conservative form' , 'Non-conservative form'});
grid on
```

Now as we the grid points in the above mentioned Nozzle main fucntion program is 31 , it is because about this about steady has been occured for both Conservative and Non-Conservative flow equations.

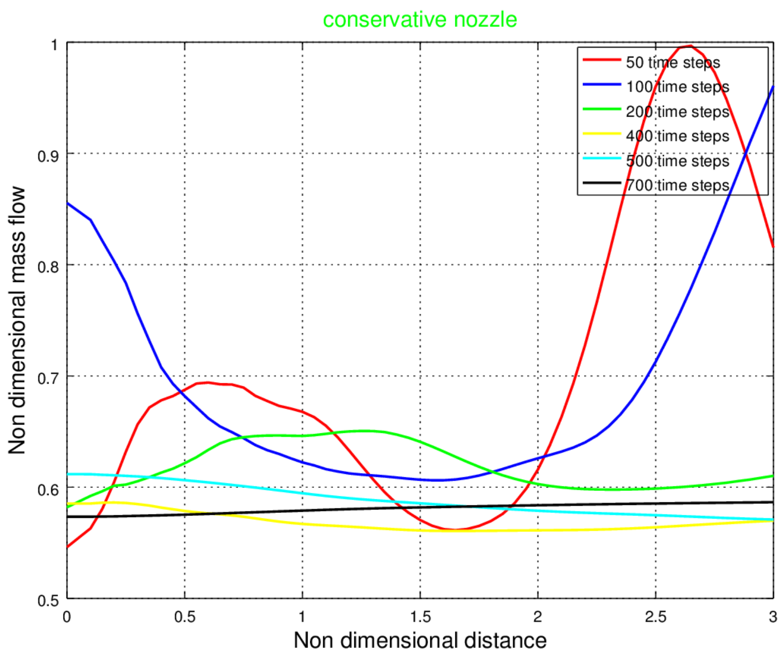
Now as we know the maximum grid points , lets check out the Grid-independence for the Conservative and Non-conservative flow program .

Grid-independence is known as changing the grid-points to check whether the Conservative and Non-conservative flow may provide the same results as before .

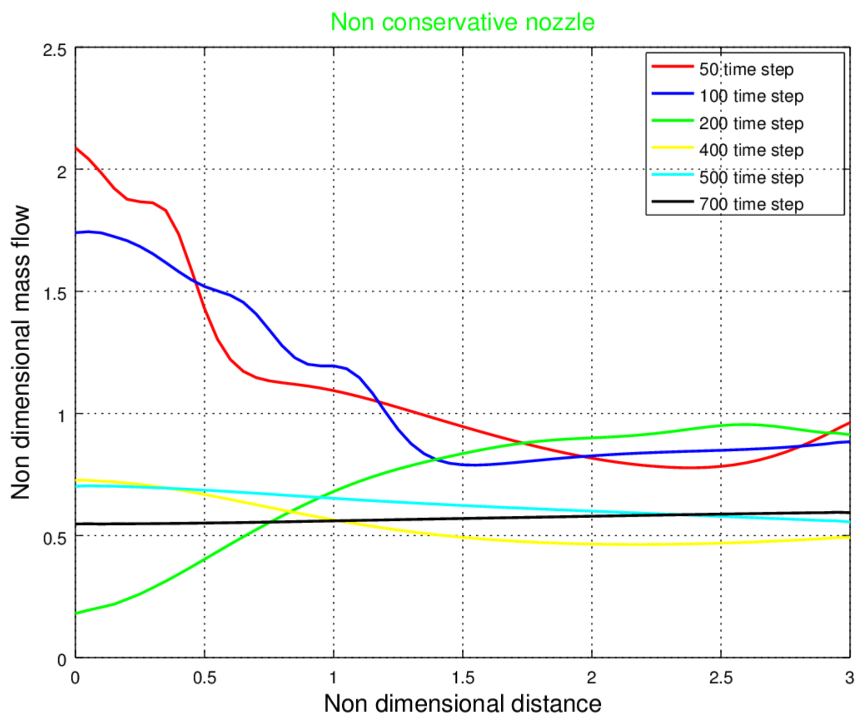
Now the Grid-points are doubled to **61 ,** and the time steps values are also increased because to obtain the elapsed Non dimensional time for both the flow maybe the same . By increasing the time steps we can carry the grid-independence by comparing the graphs obtained for both the cases.

Now let us see the output graphs for **increased Grid points** :

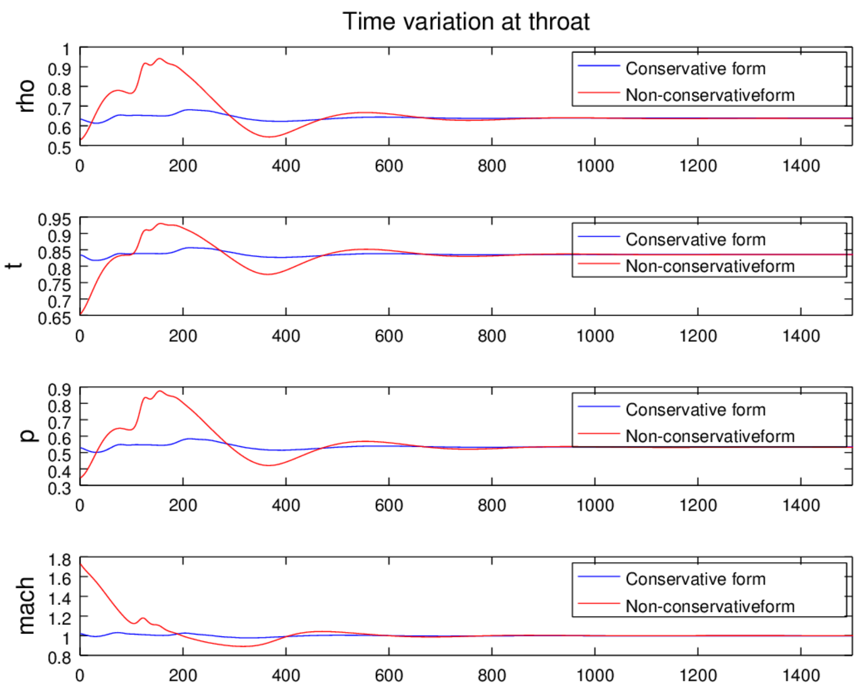
- ***Mass Flow Rate for Conervative flow :***



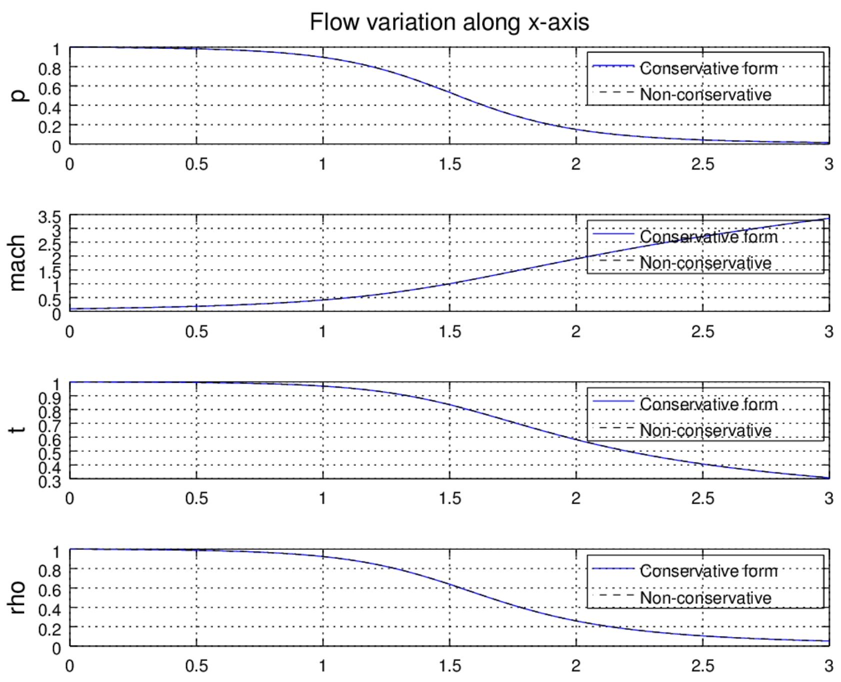
- ***Mass flow rate for Non-Conservative flow :***



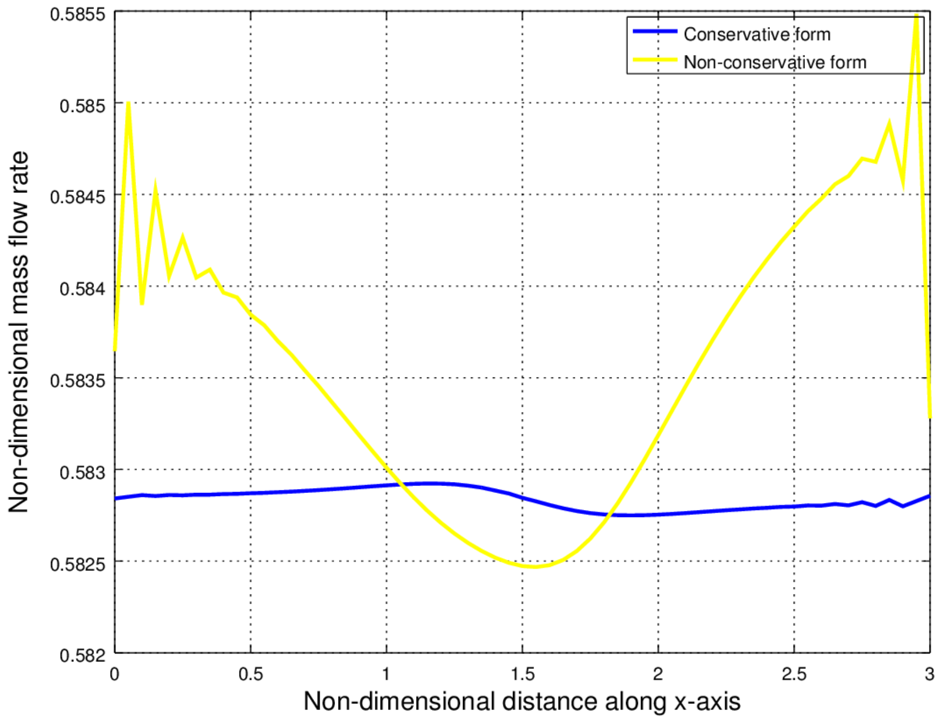
- ***Time variation at throat :***



- ***Flow variation along X-axis :***



- ***Mass flow rate comparison :***



Now we can see the difference in the graphs. We can notice that for n = 61 the steady state is obtained after 1000 time steps.

Now which method is faster :

We have made two different solutions for the equations , now we need to find which method is faster than one another.

For n = 31 ;

Conservation time = 7.9845

Non-Conservative time = 7.2293

For n =61 ;

Conservative time = 29.900

Non-Conservative time = 27.643

So, from these above data we came to a conlusion that ***Conservative forn requires more computational time than Non-Conservative form.***